Performance Exercises

Bruce Momjian

February 2004

Hardware

1. Determine the amount of RAM on your computer
2. Determine the amount of swap allocated and currently used
3. Determine the top CPU using process
4. Determine the top memory using process
5. Determine page in/page outs
6. *Determine the top I/O using process
7. Display all files opened by the PostgreSQL super-user
8. *Determine the size of your kernel's disk buffer cache
9. Find the amount of shared memory used by PostgreSQL as reported by the operating system
10. Open multiple sessions and find the pid of each session
11. *Increase shared buffers until swapping starts
12. Show a performance improvement by turning off fsync
13. Show performance difference by moving pg_xlog to another disk drive, if possible

Optimizer

1. Create queries that use nested, hash, and merge joins First do each in a separate query, and then see if you can do them all in a single query.
2. Use "SET enable_seqscan TO off" to force an index scan in a case where the optimizer chose a sequential scan and test to see if it is faster
3. Select the 'tid' column to see how many rows appear on a page for a typical table
4. *Turn off some of the optimizer enable_* flags and see if you can force a better query plan than the optimizer's default choices
5. *Adjust random page cost to force an index scan
6. Create a query that uses GEQO

Queries

1. Show improved query performance after ANALYZE
2. Generate an EXPLAIN output that is more than twenty lines
3. Find table statistics collected by ANALYZE
4. Show a query that changes its plan with different column dispersions

---

5. Decrease the free space map and illustrate VACUUM can't reuse disk space
6. Show improved performance with CLUSTER

---

7. Compare COPY vs INSERT peformance (use pg_dump)
8. Test performance of IN vs EXISTS for a subquery returning many rows
9. Show a different query plan when using LIMIT
10. *Find a query where LIMIT is faster than doing a CURSOR and fetching the first row
11. Create a functional index and write a query that uses it
12. Test the speed of PREPAREd queries
13. *Compare inline vs large object performance